Juhani Kemppinen

Spotify Web API:n käyttö web-sovelluksessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ohjelmistotekniikka

Insinöörityö

19.4.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Juhani Kemppinen Spotify Web API:n käyttö web-sovelluksessa 31 sivua 19.4.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tietotekniikan insinööri
Ammatillinen pääaine	Ohjelmistotekniikka
Ohjaajat	Yliopettaja Auvo Häkkinen

Insinöörityön tarkoituksena oli tutkia Spotify Web API:n käyttöä web-sovelluksissa. Lisäksi tavoitteena oli suunnitella ja kehittää oma web-sovellus, joka hyödyntää toiminnassaan Spotify Web API:a.

Työssä perehdyttiin API:n taustalla vaikuttavaan arkkitehtuurimalli REST:iin ja sen rajoitteisiin sekä Spotify Web API:n tekniikkaan, kuten pyyntöön, vastaukseen ja päätepisteisiin. Työssä tutkittiin myös, millä tavoin Spotify Web API:a on hyödynnetty internetistä löytyneissä web-sovelluksissa.

Lopuksi toteutettiin Sube-sovellus, jonka tarkoituksena oli seurata sille määrättyjä artisteja ja kerätä näiden uudet Spotify-julkaisut tietokantaan. Julkaisut haettiin Spotify Web API:n päätepisteitä käyttäen. Uudet julkaisut esitettiin Suben web-sivustolla genrejen perusteella eri listoihin lajiteltuina.

Työn tuloksena syntyi pätevä työkalu uuden musiikin löytämiseen Sube-sovelluksen muodossa sekä monipuolinen katsaus erilaisiin tapoihin hyödyntää Spotify Web API:a web-sovelluskehityksessä. Löytyneiden web-sovellusten perusteella pystyttiin havaitsemaan, että Spotify Web API:a käyttäviä sovelluksia löytyi verrattain vähän, ja ne olivat huomattavan samankaltaisia keskenään. Spotifyn API ei ainakaan toistaiseksi ole laajassa suosiossa web-kehityksessä.

Avainsanat	Spotify, Web API, REST, JSON, HTTP, web-ohjelmointi

Author	Juhani Kemppinen
Title	Use of Spotify Web API in web application
Number of Pages	31 pages
Date	4 April 2018
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Auvo Häkkinen, Principal Lecturer

The purpose of this thesis was to study the use of Spotify Web API in web applications. In addition, the aim was to design and develop a web application that utilizes Spotify Web API.

The study examined architecture model REST along with its constraints and technology of Spotify Web API such as requests, responses and endpoints. In addition, web applications utilizing Spotify Web API were explored.

Finally, web application Sube was implemented to track predefined artists and collect their new releases into a database. Releases were fetched using Spotify Web API endpoints. New releases were presented on the Sube website and they were sorted into lists by their genre.

As a result of this thesis a competent tool to find new music was born. The study also revealed a versatile look at different ways to utilize the Spotify Web API in web development. Based on the found web applications, it was clear the amount of web applications utilizing Spotify Web API was small. The applications were also relatively similar with each other. At least for now Spotify's API is by no means a popular in web development.

Keywords	Spotify, Web API, REST, JSON, HTTP, web programming

Lyhenteet

AJAX	Asynchronous JavaScript And XML. Joukko tekniikoita, joiden avulla web-sovellukset voivat kommunikoida palvelimen kanssa ilman sivun päivitystä.
API	Application Programming Interface. Ohjelmointirajapinta.
REST	Representational State Transfer. Ohjelmistoarkkitehtuurimalli mm. ohjelmointirajapintojen toteuttamiseen.
HTTP	Hypertext Transfer Protocol. Hypertekstin siirtoprotokolla. Käytetään tiedonsiirtoon mm. selainten ja web-palvelinten välillä.
HTTPS	HTTP:n suojattu versio. Tiedot salataan ennen lähettämistä.
JSON	JavaScript Object Notation. Tiedostomuoto.
OAuth	HTTP:tä hyödyntävä valtuutusstandardi.
SQL	Structured Query Language. Standardoitu kyselykieli, jolla voidaan hallita relaatiotietokantoja.
URI	Uniform Resource Identifier. Resurssin tunniste. Yleisin URI:n muoto on URL.
URL	Uniform Resource Locator. Osoittaa resurssin sijainnin. Yleisimmin web-osoite.

Sisällys

1 Johdanto

Musiikin kuuntelu on muuttunut 2000-luvulla merkittävästi kuluttajien siirtyessä fyysisistä formaateista verkkoon ja suoratoistopalveluihin. Teoston teettämän tutkimuksen mukaan jopa 80 % pohjoismaisista musiikin kuluttajista käyttää suoratoistopalveluita musiikin kuunteluun [1]. Yksi merkittävimmistä suoratoistopalveluista on ruotsalainen Spotify, jolla on yli 100 miljoonaa aktiivista käyttäjää ja yli 60 miljoonaa maksavaa asiakasta [2]. Spotifya käyttää arviolta viikossa noin 1,5 miljoonaa suomalaista [1].

Valtavan kappalekirjaston lisäksi Spotify tarjoaa ohjelmistokehittäjille myös kattavan ohjelmointirajapinnan, Spotify Web API:n. Sen avulla kehittäjät voivat muun muassa hakea Spotifylta dataa kappaleista, artisteista ja käyttäjistä. Rajapinta noudattaa REST-periaatteita, ja kommunikaatio sen ja asiakkaan välillä tapahtuu HTTP-pyyntöjen avulla.

Tämän insinöörityön tavoitteena on syventyä edellä mainittuun rajapintaan, tutkia, minkälaisia mahdollisuuksia se tarjoaa, sekä kehittää Spotify Web API:a hyödyntävä websovellus. Kehitettävän sovelluksen pääasiallinen tarkoitus on tarjota Spotifyn käyttäjille selkeä lista uusista Spotify-julkaisuista ennalta määrätyiltä artisteilta. Tällä hetkellä Spotifyssa ei ole ominaisuutta, jonka avulla käyttäjä näkisi helposti haluamiensa artistien uudet julkaisut, ja uskon, että tällaiselle ominaisuudelle olisi kysyntää käyttäjien keskuudessa. Sovellus tulee hyödyntämään Spotifyn ohjelmointirajapinnan lisäksi muun muassa MySQL-tietokantaa, JavaScriptiä ja Pythonia.

Työn toisessa luvussa esitellään Spotifya palveluna sekä yritystä sen takana. Kolmannessa luvussa syvennytään Spotify Web API:n noudattamaan REST-arkkitehtuurimalliin sekä API:n tekniikkaan ja toimintaan. Neljännessä luvussa tarkastellaan API:a hyödyntäviä web-sovelluksia. Työn viidennessä luvussa tutustutaan kehittämääni Sube-sovellukseen, joka hyödyntää Spotify Web API:a toiminnassaan.

2 Spotify

Spotify on ruotsalaisten Daniel Ek:in ja Martin Lorentzon vuonna 2008 perustama musiikin suoratoistopalvelu. Palvelussa on kuunneltavissa yhteensä yli 35 miljoonaa kappaletta [2]. Palvelua kehittää Tukholmassa pääkonttoriaan pitävä yritys Spotify AB. Yritys työllistää yli 1600 täyspäiväistä työntekijää, ja sen liikevaihto ylitti 2 miljardia Yhdysvaltain dollaria vuonna 2015 [4].

Spotify toimii ns. freemium-mallin mukaan, jossa ilmaiskäyttäjälle tarjotaan vain perustoiminnot mainoksilla ja maksavalle tilaajalle tarjotaan lisäominaisuuksia ilman mainoksia, kuten esimerkiksi musiikin parempi äänenlaatu ja kappaleiden latausmahdollisuus.

Palvelulla on noin 71 miljoonaa tilaajaa ja noin 159 miljoonaa kuukausittaista aktiivista käyttäjää [5]. Spotify on saatavilla tietokoneille ladattavana sovelluksena ja web-pohjaisena. Sovellus on saatavilla myös puhelimille, tableteille ja monille muille laitteille.

Musiikin kuuntelu on viimeisen vuosikymmenen aikana siirtynyt fyysisistä formaateista lähes kokonaan verkkoon. Teoston teettämän tutkimuksen mukaan yli 80 % suomalaisista kuuntelee musiikkia suoratoistopalveluiden kautta [1]. Yksi merkittävimpiä tekijöitä tässä kehityksessä ovat olleet helppokäyttöiset suoratoistopalvelut, kuten esimerkiksi Spotify, Deezer ja Apple Music.

Spotify on 71 miljoonalla maksavalla premium-käyttäjällään musiikin suoratoistopalveluiden kiistaton markkinajohtaja. Spotifyn suurin kilpailija on Apple Music 36 miljoonalla tilaajallaan. Muita varteenotettavia kilpailijoita Spotifylle ovat muun muassa Amazon noin 16 miljoonalla tilaajallaan ja Deezer noin 6 miljoonalla tilaajallaan [6]. Vaikka ero Spotifyn ja Apple Musicin kokonaistilaajamäärien välillä on vielä verrattain suuri, Apple Musicin tilaajamäärä kasvaa Yhdysvalloissa noin 5 % kuukaudessa, kun Spotifyn tilaajamäärä kasvaa samalla alueella vain noin 2 % kuukaudessa. Mikäli kasvu pysyy samana, Apple Music ohittaa Spotifyn yhdysvaltalaistilaajien määrässä kesän 2018 aikana [7].

Käyttäjän kannalta Spotify ja Apple Music eivät eroa merkittävästi toisistaan. Molempien musiikkikirjasto kattaa yli 35 miljoonaa kappaletta (Apple Musicin jopa 45 miljoonaa) ja sovellus on saatavilla kaikille yleisimmille tietoteknisille laitteille, joskin Spotify hieman suuremmalle määrälle. Spotify tarjoaa käyttäjilleen myös mahdollisuuden käyttää palvelua web-selaimen kautta toisin kuin Apple Music, joka vaatii sovelluksen toimiakseen.

Palvelut maksavat täsmälleen saman verran lukuun ottamatta mainoksia sisältävää ilmaisvaihtoehtoa, jota ainoastaan Spotify tarjoaa. Apple Musicin merkittävin etu Spotifyhin verrattuna on sen sulautuvuus muihin Applen tuotteisiin. Apple Musicin avulla käyttäjällä on mahdollisuus kuunnella Applen iCloud Music Libraryyn tallentamiaan kappaleita missä tahansa. Sovellus on myös esiasennettuna useisiin Applen laitteisiin, mikä houkuttelee helposti uusia käyttäjiä sovelluksen käyttäjäksi ja tilaajaksi.

3 Spotify Web API

Spotify Web API on Spotifyn ohjelmointirajapinta, joka mahdollistaa keskustelun sovellusten ja Spotifyn palvelinten välillä. Rajapinnan avulla sovellukset voivat esimerkiksi hakea metadataa kappaleista, artisteista tai soittolistoista sekä muokata käyttäjän soittolistoja.

Rajapinta on REST-pohjainen. Se hyödyntää HTTP-protokollaa tiedonsiirrossa. Web API:n perusosoite on https://api.spotify.com. API-pyynnöissä perusosoitteen perään liitetään se päätepiste (engl. endpoint), jolta dataa halutaan hakea. Päätepisteet toimivat API:n yhteyspisteinä.

Tammikuusta 2017 lähtien Spotify on vaatinut, että kaikki API-pyynnöt ovat autentikoituja. Tämä tarkoittaa sitä, että jokaisen pyynnön mukaan on liitettävä käyttöoikeustietue (engl. access token). Käyttöoikeustietueen avulla palvelin tietää, mikä sovellus ja käyttäjätunnus pyynnön on lähettänyt. Spotify Web API käyttää valtuutukseen OAuth-valtuutusprotokollaa. Valtuutusta ja OAuth-protokollaa käsitellään tarkemmin luvussa 3.4.

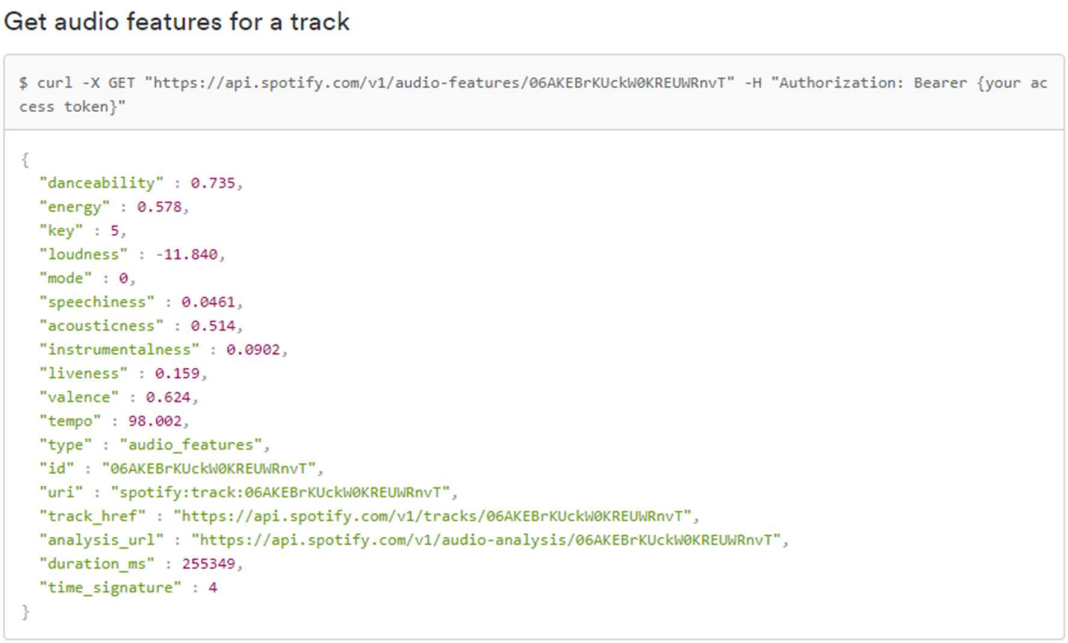
3.1 REST

REpresentational State Transfer eli REST on arkkitehtuurimalli hajautettujen järjestelmien suunnitteluun. REST:n perusajatuksena on hyödyntää WWW:n jo olemassa olevia teknologioita ja protokollia, kuten URI:a ja HTTP:tä. REST nojaa myös vahvasti asiakas-palvelin-arkkitehtuuriin, jossa asiakkaan (esimerkiksi Web API:a käyttävä sovellus) ja palvelimen (esimerkiksi Web API) väliset tehtävät erotetaan selvästi toisistaan. [3.]

REST koostuu monista rajoitteista ja säännöistä. Kuusi näistä rajoitteista ovat ohjaavia rajoitteita, joita REST-pohjaisen järjestelmän on pakko noudattaa [8]. Nämä ovat asiakas-palvelin-arkkitehtuuri, tilattomuus, välimuisti, yhdenmukainen rajapinta, kerroksittainen järjestelmä ja code-on-demand. Näiden rajoitteiden tarkoitus on parantaa rajapintojen suorituskykyä, skaalautuvuutta, yksinkertaisuutta ja muunneltavuutta. [3.]

REST:n käyttö arkkitehtuurimallina näkyy Spotifyn API:ssa erittäin selvästi ja rajoitteita noudatetaan kiitettävästi. Asiakas ja API kommunikoivat HTTP-protokollan ja sen metodien avulla [9]. API ei säilö istuntotietoja asiakkaasta ja jokainen API-kutsu käsitellään yksitellen. Kaikki API:n palauttama data on myös JSON-muodossa.

Kaikki kommunikaatio asiakkaan ja API:n välillä tapahtuu HTTP-pyynnöillä. Asiakas pyytää HTTP-metodien avulla palvelinta tekemään erinäisiä asioita Spotifyn resursseille ja palvelin palauttaa asiakkaalle esityksen resurssista JSON-muodossa [9]. Kuvassa 1 on esimerkki Spotify Web API:n palauttamasta vastauksesta JSON-muodossa.

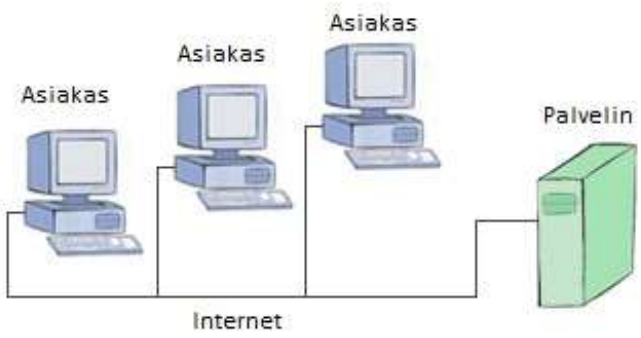


Kuva 1. Esimerkki Spotify Web API:n palauttamasta JSON-muotoisesta vastauksesta.

3.1.1 Asiakas-palvelin-arkkitehtuuri

Asiakas-palvelin-arkkitehtuurin keskeisenä ajatuksena on erottaa palvelin ja asiakas selkeästi toisistaan niin, että molemmilla on omat tehtävänsä ja huolenaiheensa. Tämä

edesauttaa järjestelmän hajauttamista ja mahdollistaa sen, että molempia voidaan päivittää ja kehittää itsenäisesti. Palvelimen ja asiakkaan erottamisella myös parannetaan järjestelmän skaalautuvuutta. [3.] Kuvassa 2 on esitetty asiakas-palvelin-arkkitehtuurin toimintaperiaate.



Kuva 2. Asiakas-palvelin-arkkitehtuurin toimintaperiaate.

Palvelimen tehtävänä on tarjota Unique Resource Identifier eli URI. URI toimii resurssin osoitteena. Sen avulla asiakas pääsee yhteen tai useampaan resurssiin käsiksi ja voi HTTP-pyynnön avulla pyytää palvelinta tekemään toimenpiteitä resurssille. Palvelin joko hylkää tämän pyynnön tai tekee pyydetyt toimenpiteet ja lähettää asiakkaalle vastauksen. [10.]

3.1.2 Tilattomuus

REST:n toinen ehdoton rajoite on nimeltään tilattomuus, joka vaatii, että kommunikaatio palvelimen ja asiakkaan välillä on tilatonta. Palvelin ei tallenna mitään istuntotietoja asiakkaasta tai sen aiemmista pyynnöistä ja täten jokaisen pyynnön täytyy sisältää kaikki tarvittava tieto pyynnön ymmärtämiseksi. Istuntotiedot säilytetään asiakkaalla. Rajoite parantaa järjestelmän skaalautuvuutta ja luotettavuutta. [3.]

3.1.3 Välimuisti

Kolmas rajoite vaatii palvelinta liittämään jokaiseen vastaukseen tiedon siitä, voidaanko vastauksen data tallentaa välimuistiin joko asiakkaan päässä tai muualla asiakkaan ja palvelimen välillä. Välimuistiin tallentamisella voidaan eliminoida toistuvia pyyntöjä palvelimelle, joissa vastauksen sisältö ei ole todennäköisesti muuttunut. Tällä saadaan parannettua sekä asiakkaan että palvelimen tehokkuutta. [3.]

3.1.4 Yhdenmukainen rajapinta

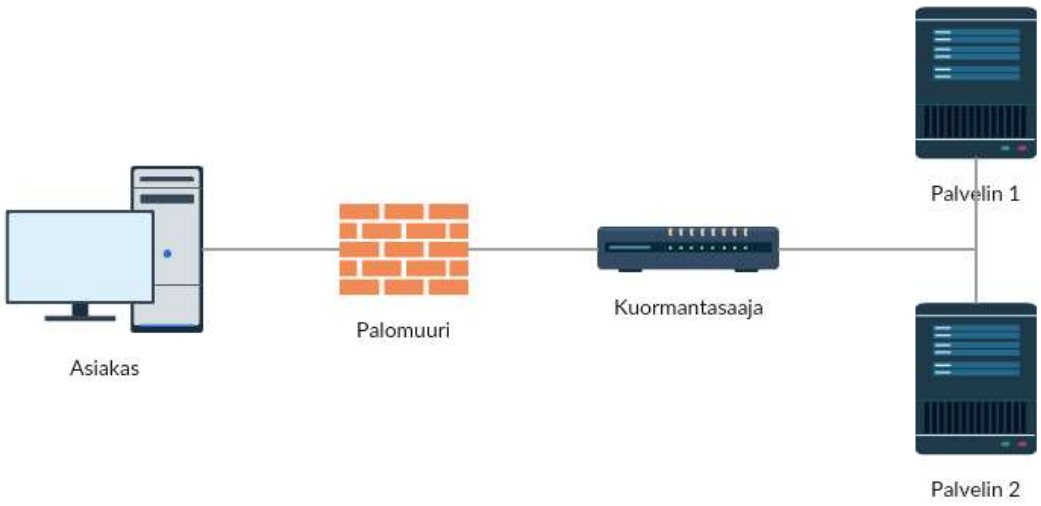
Yhdenmukainen rajapinta -rajoitteen tarkoituksena on yksinkertaistaa järjestelmää ja erottaa sen osat niin, että kukin osa voi kehittyä itsenäisesti. Rajoitteen haittapuolena on järjestelmän tehokkuuden aleneminen, koska kaikki tieto pitää siirtää aina samassa muodossa sen sijaan, että se siirrettäisiin kunkin palvelun kannalta parhaassa muodossa. Rajoite koostuu neljästä alirajoitteesta: resurssien tunnistamisesta, resurssien manipuloinnista esitysten (engl. representation) kautta, itseselitteisistä viesteistä ja hypermediasta sovelluksen tilakoneena. [3.]

Ymmärtääksemme paremmin tämän rajoitteen on ensin ymmärrettävä, mikä REST:n mukaan on resurssi. REST:ssä resurssilla tarkoitetaan mitä tahansa nimettyä kokonaisuutta, jota palvelin tarjoaa asiakkaalle. Resursseja ovat esimerkiksi WWW-sivu, dokumentti tai tietokannan taulussa oleva rivi [3].

Rajoitteen mukaan palvelulle tulevissa pyynnöissä tulee viitata, mihin palvelun resurssiin pyyntö kohdistuu. Web-pohjaisissa REST-palveluissa resurssiin viitataan useimmiten URI:lla. Palvelu luo pyydetystä resurssista esityksen, jonka se palauttaa asiakkaalle. Tämän esityksen tulee pitää sisällään kaikki tarvittava tieto resurssin manipulointiin, kuten sen muokkaamiseen tai poistamiseen. Itseselitteisillä viesteillä tarkoitetaan sitä, että jokaisen palvelimen ja asiakkaan välisen viestin täytyy pitää sisällään kaikki tarvittava tieto viestin käsittelyyn, eikä lisätietoja tarvitse kysyä. Viimeinen alirajoite kuvaa tapaa, jolla sovelluksen tila muuttuu. Rajoite vaatii, että REST-pohjaisen palvelun tulee olla hypermedia-pohjainen. Käytännössä tämä tarkoittaa sitä, että asiakkaan täytyy tietää palvelusta mahdollisimman vähän etukäteen ja palvelin sisällyttää vastaukseensa linkit muihin asiakkaalle hyödyllisiin resursseihin. [3.]

3.1.5 Kerroksittainen järjestelmä

Kerroksittaisen järjestelmän ideana on jakaa järjestelmä hierarkkisiin kerroksiin, jotka kommunikoivat ainoastaan itseään välittömästi ylä- ja alapuolella olevien kerrosten kanssa ja näin estävät monimutkaisten suhteiden syntymistä palvelinten välille. Asiakkaan ja palvelimen välille voidaan lisätä välikerroksia, jotka pitävät sisällään esimerkiksi palomuureja tai kuormantasausta (engl. load balancer). Välikerroksissa sijaitsevat palvelimet voivat muun muassa siepata asiakkaan ja palvelimen välistä liikennettä ja tallentaa sen välimuistiin. [3.] Kuvassa 3 on esimerkki kerroksittaisesta järjestelmästä.



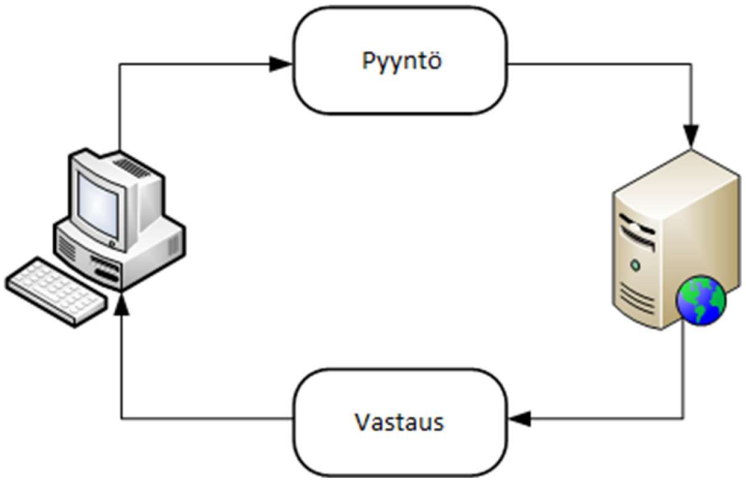
Kuva 3. Esimerkki kerroksittaisesta järjestelmästä.

3.1.6 Code on demand

Viimeinen REST:n kuudesta ohjaavasta rajoitteesta mahdollistaa asiakassovelluksen toiminnallisuuden laajentamisen lataamalla ja suorittamalla koodin asiakkaan päässä. Rajoitteella yksinkertaistetaan asiakassovellusta vaatimalla vähemmän etukäteen toteutettuja ominaisuuksia. Lataamalla koodia saadaan parannettua järjestelmän laajennettavuutta, mutta järjestelmän näkyvyys kärsii. Tästä syystä tämä rajoite on vapaaehtoinen. [3.]

3.2 Pyyntö, vastaus ja päätepisteet

Spotify Web API (SWA) hyödyntää REST-periaatteiden mukaisesti HTTP-protokollaa ja sen metodeja tiedonsiirrossa. Kaikki kommunikaatio asiakkaan ja Spotifyn palvelimen välillä rakentuu HTTP-pyyntöjen ja vastausten ympärille. Palvelin kuuntelee useaa eri päätepistettä tulevien pyyntöjen varalta ja on valmiina käsittelemään pyynnöt sekä palauttamaan vastauksen. Pyynnön ja vastauksen perusperiaate on esitetty kuvassa 4. Yhteensä Spotify hyödyntää päätepisteillään neljää HTTP-metodia, GETiä, POSTia, PUTia sekä DELETEä. GET-metodilla pyydetään dataa resurssista, POSTilla ja PUTilla lähetetään dataa resurssiin, ja DELETEllä poistetaan resurssi [9].



Kuva 4. Pyyntö ja vastaus. Asiakas tekee pyynnön ja palvelin palauttaa vastauksen.

API-pyyntö koostuu päätepisteen määräämästä HTTP-metodista, resurssin URL:sta, header-kentästä, sekä joillain päätepisteillä myös query-parametrista. Alapuolella on yksinkertainen esimerkki API-pyynnöstä https://api.spotify.com/**v1/artists/{id}** -päätepisteelle. URL:ssa oleva merkkijono ”0OdUWJ0sBjDrqHygGUXeCF” on Spotifyn sisäinen-ID artistille ja ”-H”:n perässä oleva parametri on header-kenttä. /v1/artists/{id}-päätepiste ei käytä query-parametria, joten sitä ei tässä pyynnössä ole. Header-kentässä on ainoastaan käyttöoikeustietue käyttäjän valtuuttamiseksi. Perehdymme valtuuttamiseen ja autentikointiin tarkemmin luvussa 3.4.

```
GET "https://api.spotify.com/v1/artists/0OdUWJ0sBjDrqHygGUXeCF" -H "Authoriza
tion: Bearer {your access token}"
```

Spotify on määritellyt tarkasti kehyksen, jota API-pyyntöjen täytyy noudattaa, eikä se jätä tulkinnan varaa asiakkaalle. API:n dokumentaatiossa on määritelty jokaiselle päätepisteelle HTTP-metodit sekä parametrit, jotka sille tulevissa pyynnöissä täytyy olla. Dokumentaatio tarjoaa myös esimerkkipyynnön jokaiselle päätepisteelle [11].

Vastaus sisältää aina vähintään HTTP-status-koodin, joka kertoo asiakkaalle pyynnön tilan. 2:lla alkavat koodit (2XX) tarkoittavat, että pyyntö on käsitelty ja suoritettu onnistuneesti. 4:llä alkavat koodit (4XX) kertovat, että pyynnön käsittelyssä on tapahtunut jokin virhe, joka johtuu asiakkaasta. Koodi 404 esimerkiksi kertoo, että pyydettyä resurssia ei löydy, eli pyynnössä välitetty resurssin URI on virheellinen. 5XX-koodit tarkoittavat, että pyynnön käsittelyssä on tapahtunut virhe ja vika on palvelimen. [9.]

GET-pyyntöjä vastaanottavia päätepisteitä käytetään tyypillisesti datan hakemiseen palvelimelta. Spotifyn API palauttaa näistä päätepisteistä asiakkaalle muutamaa päätepistettä lukuunottamatta vastauksen, joka pitää sisällään JSON-muotoisen olion, eli käytännössä kokoelman nimi-arvo-pareja, jotka liittyvät pyydettyyn resurssiin.

Otetaan esimerkiksi Spotify Web API:n päätepiste https://api.spotify.com**/v1/tracks/{id}**. Päätepisteen tehtävä on palauttaa tietoa jostain tietystä kappaleesta. Se ottaa vastaan ainoastaan GET-pyyntöjä ja vaatii, että asiakas tietää haetun kappaleen Spotify ID:n, joka lisätään päätepisteen URL:n perään. Vastauksessaan se palauttaa asiakkaalle JSON-olion pyydetystä kappaleesta, joka pitää sisällään kappaleen kannalta oleellisia nimi-arvo-pareja, kuten kappaleen nimen, artistin nimen, kappaleen pituuden ja linkin kappaleeseen. Parit vaihtelevat resurssin ja päätepisteen mukaan, mutta data palautetaan aina JSON:in nimi-arvo-pareissa. [12.]

Päätepisteet (engl. endpoint) ovat keskeinen osa Spotify Web API:n sekä muiden Web API:en toimintaa. Ne ovat palvelimen tarjoamia yhteyspisteitä, jotka ottavat vastaan asiakkaiden lähettämiä API-pyyntöjä ja palauttavat asiakkaille vastauksen. Päätepisteet toimivat resurssien osoitteina, samaan tapaan kuin WWW:stä tutut web-osoitteet toimivat web-sivujen osoitteina. Päätepisteet eivät itse tee pyyntöjä tai aloita kommunikaatiota, vaan ne ainoastaan vastaanottavat pyyntöjä. Ne ovat ainoa tiedonsiirtoväylä asiakkaan ja Spotify Web API:n välillä.

Yhteensä Spotifylla on yli 50 päätepistettä API:ssaan ja jokaisella näistä on oma tehtävänsä. Spotify lisää päätepisteitä jatkuvasti rajapintaansa. Valtaosa Spotify Web API:n päätepisteistä ottaa vastaan vain tietyn HTTP-metodin sisältäviä pyyntöjä, mutta joillekin voi lähettää GET-, PUT- ja DELETE-pyyntöjä. Esimerkiksi https:// api.spotify.com/**v1/me/following** -päätepisteeltä voi GET-pyynnöllä hakea käyttäjän seuraamat artistit, PUT-pyynnöllä lisätä artistin seurattuihin ja DELETE-pyynnöllä poistaa artistin seuratuista. [11.]

Päätepisteet voidaan jakaa karkeasti kolmeen ryhmään niiden toiminnallisuuden perusteella. Ensimmäisen ryhmän päätepisteet palauttavat julkista katalogidataa, kuten musiikin metadataa. Toisen ryhmän päätepisteet palauttavat ja muokkaavat käyttäjätilikohtaista dataa, kuten soittolistojen sisältöä. Kolmas ryhmä palauttaa ja muokkaa käyttäjän laitteen soiton tilaa.

3.3 Näkyvyysalue

Näkyvyysalueella (engl. scope) määritellään sovelluksen käyttöoikeuden laajuus käyttäjän dataan. Näkyvyysalue koostuu käyttöoikeuslohkoista, joiden avulla sovelluksen käyttöoikeus voidaan rajata hyvin tarkasti. Sovellus pyytää käyttäjää hyväksymään ainoastaan sellaisen näkyvyysalueen, jota se tarvitsee. Näin käyttäjä tietää tarkalleen, mihin tietoihin sovelluksella on pääsy.

Näkyvyysalue hyväksytetään käyttäjällä valtuuttamisen yhteydessä. Hyväksyttämisen jälkeen Spotify Web API räätälöi hyväksytyn alueen pohjalta käyttöoikeustietueen, jolla on pääsy vain käyttäjän valtuuttamaan dataan. Käyttöoikeustietueella, joka ei sisällä näkyvyysaluetta, on vakiona lukuoikeus vain julkisesti saatavilla olevaan dataan. Julkista dataa on kaikki, mikä on käyttäjätilistä näkyvillä muille Spotifyn käyttäjille esimerkiksi työpöytäsovelluksen kautta.

Spotify Web API:lla on yhteensä 18 käyttöoikeuslohkoa. Näistä 11 myöntää sovellukselle lukuoikeuden käyttäjän yksityiseen dataan, kuten esimerkiksi sähköpostiosoitteeseen tai kuunnelluimpiin artisteihin. Neljä lohkoa myöntää muokkausoikeuden esimerkiksi soittolistoihin. Kaksi lohkoa antaa sovellukselle oikeuden hallita musiikin toistoa käyttäjän laitteilla ja yksi oikeuttaa sovelluksen lataamaan kuvia palveluun käyttäjän puolesta. [13.]

3.4 Valtuutus

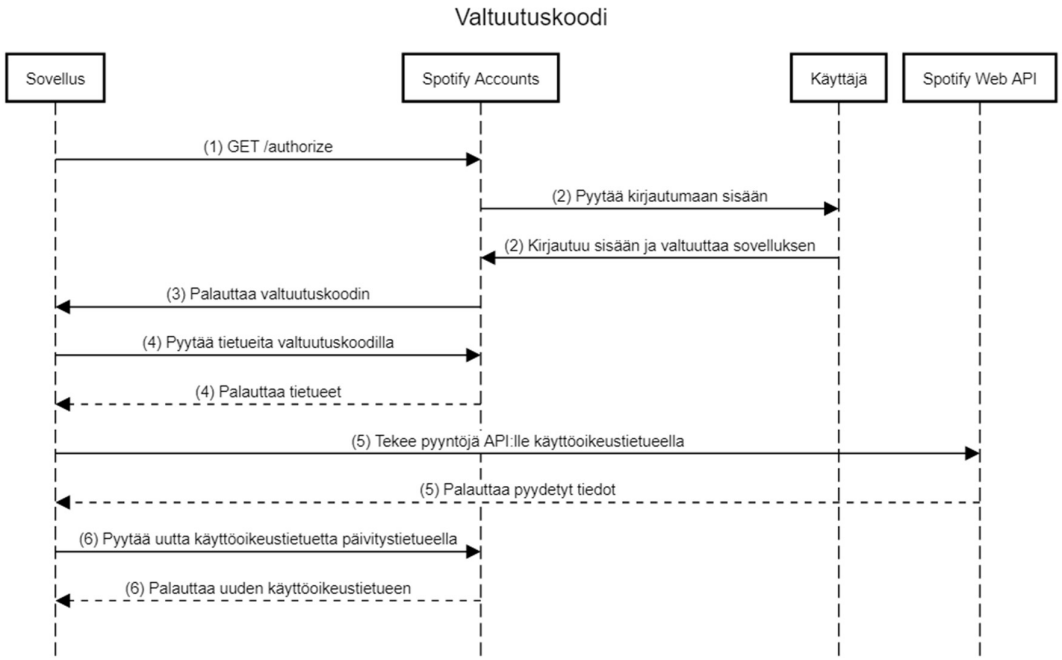
Spotify vaatii paitsi musiikkipalvelun, myös Web API:n käyttäjiltä rekisteröitymistä. Näin ollen jokaisen API-pyynnön täytyy sisältää käyttöoikeustietue, jonka avulla Spotify pystyy liittämään pyynnön johonkin olemassa olevaan käyttäjätunnukseen. Spotify Web API hyödyntää valtuutuksessa OAuth 2.0 -standardia. Ensimmäinen askel käyttöoikeustietueen saamiseen on rekisteröidä API:a käyttävä asiakassovellus Spotifyn Developer-portaalissa. Rekisteröinnin yhteydessä sovellukselle generoituu asiakas-id (engl. client id) sekä asiakas-salasana (engl. client secret), joita tarvitaan seuraavissa vaiheissa käyttöoikeustietueen saamiseen. [14.]

Kun asiakas-id ja -salasana ovat generoituneet, on käyttäjällä kolme eri vaihtoehtoa valtuuttamisen viemiseksi loppuun. Spotify tarjoaa asiakkaille kolme OAuth 2.0 :n valtuutusvirtaa (engl. authorization flow), joista asiakas voi valita parhaiten itselleen ja sovellukselleen sopivan vaihtoehdon. Valtuutusvirrat ovat prosesseja, joiden päämääränä on

tunnistaa asiakas ja generoida tälle käyttöoikeustietue ja näin valtuuttaa asiakas käyttämään API:a.

3.4.1 Valtuutuskoodi

Valtuutuskoodi (engl. authorization code) on yksi kolmesta Spotifyn tukemasta valtuutusvirrasta. Se sopii parhaiten pitkäkestoisille sovelluksille, jotka hyödyntävät käyttäjän tilikohtaista dataa ja vaativat käyttäjää kirjautumaan sisään omalla Spotify-tunnuksellaan. Valtuutuskoodin toiminta havainnollistetaan sekvenssikaavion avulla kuvassa 5.



Kuva 5. Sekvenssikaavio valtuutuskoodin toiminnasta. Viesteissä olevat numerot viittaavat seuraavan luettelon vaiheisiin.

1. Sovellus pyytää valtuutusta. Palvelin tekee GET-pyynnön Spotify Accounts-palvelulle. Pyyntö pitää sisällään useita parametreja, kuten aiemmin generoidun asiakas-id:n ja uudelleenohjausosoitteen. Pyyntö voi sisältää myös näkyvyysalueen, jolla määritellään, kuinka paljon tietoa käyttäjän tilistä halutaan.

2. Palvelin pyytää käyttäjää kirjautumaan sisään Spotify-tunnuksellaan web-selaimessa. Tämän jälkeen käyttäjää pyydetään antamaan sovellukselle lupa päästä näkemään käyttäjän tilikohtaiset tiedot. Tietojen laajuus riippuu edellisen

GET-pyynnön näkyvyysalueesta. Mikäli näkyvyysaluetta ei ole määritelty aiemmassa pyynnössä, sovellus pyytää lupaa nähdä vain tilistä julkisesti saatavilla olevat tiedot.

3. Lupapyynnön hyväksymisen tai hylkäämisen jälkeen palvelin ohjaa käyttäjän aiemmin määriteltyyn uudelleenohjausosoitteeseen, joka on tyypillisesti sovelluksen web-osoite. Jos käyttäjä hyväksyi lupapyynnön, lisätään uudelleenohjausosoitteen perään valtuutuskoodi sekä tila. Seuraavassa on esimerkki uudelleenohjausosoitteesta, jonka perään on liitetty valtuutuskoodi sekä tilan arvo.

```
http://example.com/callback?code=NApCCgBkWtQjwevWrWR&state=profile%2Factivity
```

Jos käyttäjä hylkää lupapyynnön, lisätään uudelleenohjausosoitteen perään virhekoodi sekä tila. Tämän jälkeen valtuutusprosessi päättyy epäonnistuneena.

4. Valtuutuskoodin saamisen jälkeen palvelimen täytyy tehdä POST-pyyntö Spotify Accounts-palvelulle, tällä kertaa sen /api/token-päätepisteelle. Pyyntö pitää sisällään valtuutuskoodin, uudelleenohjausosoitteen, asiakas-id:n, asiakas-salasanan sekä valtuutusvirran tyypin. Pyynnön ollessa ok, Spotify Accounts-palvelu palauttaa vastauksen, joka sisältää käyttöoikeustietueen sekä päivitystietueen (engl. refresh token). Käyttöoikeustietue on voimassa yhden tunnin.

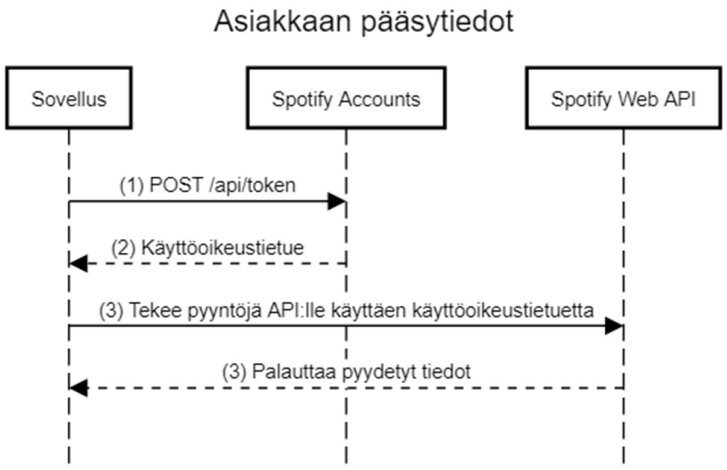
5. Käyttöoikeustietueen saamisen jälkeen palvelin voi tehdä pyyntöjä Spotify Web API:lle. Käyttöoikeustietue tulee sisällyttää jokaisen API:lle tehdyn pyynnön header-kenttään Authorization-parametrilla.

6. Käyttöoikeustietueen vanhetessa palvelin saa uuden tekemällä POST-pyynnön Accounts-palvelulle päätepisteeseen /api/token. Tällä kertaa pyyntöön tulee liittää mukaan aiemmin saatu päivitystietue. Accounts-palvelu palauttaa vastauksen, joka sisältää uuden käyttöoikeustietueen. [14.]

3.4.2 Asiakkaan valtuustiedot

Asiakkaan valtuustiedot (engl. client credentials) on kolmesta valtuutusvirrasta suppein. Se on tarkoitettu sovelluksille, jotka eivät tarvitse käyttäjätietoja ja hyödyntävät ainoas-

taan julkista dataa. Valtuutus tapahtuu palvelimelta palvelimelle, eikä loppukäyttäjä osallistu prosessiin. Asiakkaan pääsytiedot -valtuutusvirran toiminta havainnollistetaan kuvassa 6.



Kuva 6. Sekvenssikaavio asiakkaan pääsytiedot -valtuutusvirran toiminnasta. Viesteissä olevat numerot viittaavat seuraavan luettelon vaiheisiin.

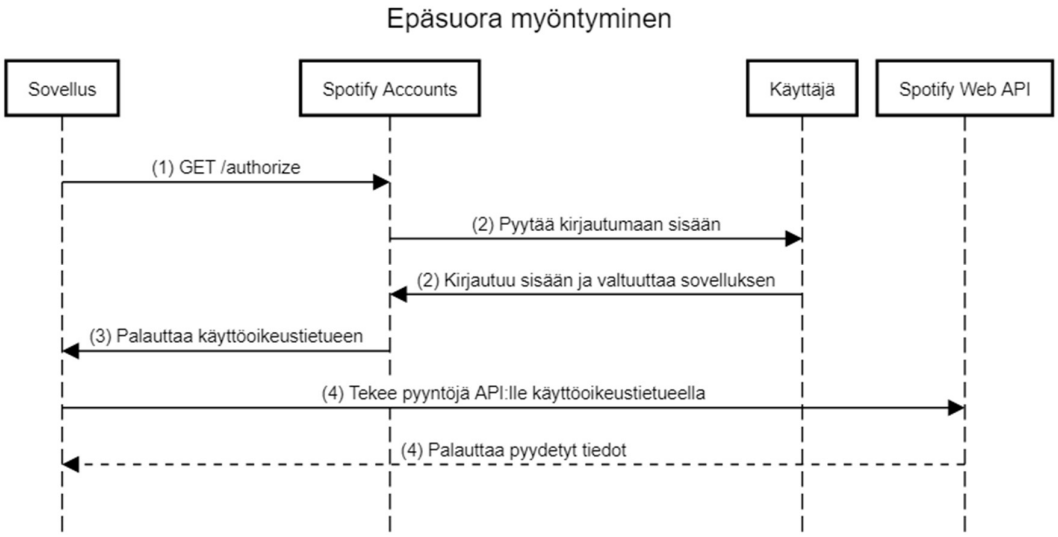
1. Palvelin tekee POST-pyynnön Spotify Accounts -palvelulle päätepisteeseen /api/token. Pyyntö sisältää valtuutusvirran tyypin ”client_credentials”, asiakas-id:n ja asiakas-salasanan.

2. Accounts-palvelu palauttaa JSON-muotoisen vastauksen, joka sisältää käyttöoikeustietueen. Käyttöoikeustietue on voimassa yhden tunnin.

3. Palvelin voi tehdä API-pyyntöjä sellaisiin päätepisteisiin, jotka palauttavat vain julkista dataa, kuten metatietoja kappaleista tai artisteista. API-pyyntöjä ei voi tehdä päätepisteisiin, jotka vaativat käyttäjän todennusta. [14.]

3.4.3 Epäsuora myöntyminen

Epäsuora myöntyminen (engl. implicit grant) on kolmas Spotify Web API:n tukemasta valtuutusvirrasta. Se on tarkoitettu JavaScript-pohjaisille selainsovelluksille. Valtuutus tapahtuu kokonaisuudessaan loppukäyttäjän selaimessa. Tästä johtuen sovelluksen asiakas-salasanaa ei käytetä tässä valtuutusvirrassa. Epäsuora myöntyminen -valtuutusvirran toiminta havainnollistetaan kuvassa 7.



Kuva 7. Sekvenssikaavio epäsuora myöntyminen -valtuutusvirran toiminnasta. Viesteissä olevat numerot viittaavat seuraavan luettelon vaiheisiin.

1. Sovellus ohjaa käyttäjän Spotify Accounts-palvelun /authorize-päätepisteeseen GET-pyynnöllä. Pyynnön mukana Accounts-palvelulle kulkeutuu sovelluksen asiakas-id, uudelleenohjausosoite (sovelluksen web-osoite) sekä muita parametreja. Parametrit liitetään pyyntöön URL:n perään, kuten alla olevassa esimerkissä.

```
https://accounts.spotify.com/authorize?client_id=5fe01282e94241328a84e7c5cc16
9164&redirect_uri=http:%2F%2Fexample.com%2Fcallback&scope=user-read-private%2
0user-read-email&response_type=token&state=123
```

2. Käyttäjää pyydetään kirjautumaan sisään Spotify-tunnuksellaan. Tämän jälkeen käyttäjää pyydetään antamaan sovellukselle lupa päästä näkemään käyttäjän tilikohtaiset tiedot. Tietojen laajuus riippuu edellisen GET-pyynnön näkyvyysalueesta. Mikäli näkyvyysaluetta ei ole määritelty aiemmassa pyynnössä, sovellus pyytää lupaa nähdä vain tilistä julkisesti saatavilla olevat tiedot.

3. Lupapyynnön hyväksymisen tai hylkäämisen jälkeen käyttäjä ohjataan aiemmin määriteltyyn uudelleenohjausosoitteeseen. Mikäli käyttäjä antoi luvan tietojen käyttämiseen, lisätään uudelleenohjausosoitteen perään käyttöoikeustietue sekä muita parametreja. JavaScript-sovellus voi näin poimia käyttöoikeustietueen

URL:sta ja käyttää sitä API-pyynnöissä. Käyttäjän hylättyä lupapyynnön uudelleenohjausosoitteen perään lisätään virhekoodi, eikä käyttöoikeustietuetta myönnetä.

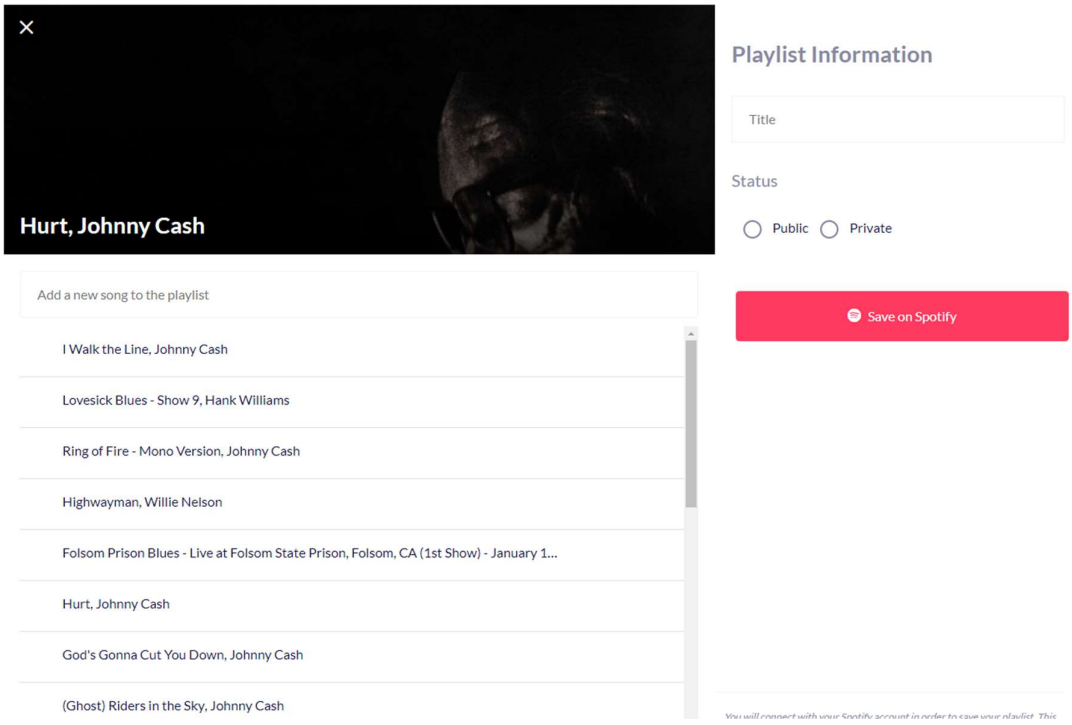
4. Käyttöoikeustietueen saamisen jälkeen selainsovellus voi tehdä pyyntöjä Spotify Web API:lle. Käyttöoikeustietue tulee sisällyttää jokaisen API:lle tehdyn pyynnön header-kenttään Authorization-parametrilla. [14.]

4 Spotify Web API:a käyttäviä sovelluksia

Tässä luvussa tutustutaan erilaisiin sovelluksiin, jotka hyödyntävät Spotify Web API:a toiminnassaan. Luvussa tutustutaan yhteensä neljään web-sovellukseen, jotka ovat nimeltään Magic Playlist, Album Availability, Spotlistr sekä Boil The Frog. Sovellukset hyödyntävät muun muassa Spotify Web API:n search-, related-artists ja albums-päätepisteitä.

4.1 Magic Playlist

Magic Playlist on Joel Loveran kehittämä web-sovellus, jonka pääasiallinen tehtävä on generoida kokonaisia Spotify-soittolistoja käyttäjän syöttämän yhden kappaleen perusteella. Käyttäjällä on myös mahdollisuus kirjautua sovellukseen sisään Spotify-tunnuksellaan ja tallentaa soittolista tunnukselleen. Kuvassa 8 on ruutukaappaus sovelluksen luomasta soittolistasta Johnny Cashin Hurt-kappaleen pohjalta.



Kuva 8. Magic Playlist -sovelluksen luoma soittolista Johnny Cashin Hurt-kappaleen perusteella. Kirjautuneella käyttäjällä on mahdollisuus tallentaa soittolista tunnukselleen sivun oikeasta reunasta.

Soittolistaa luodessaan sovellus tekee API-pyyntöjä kolmelle Spotify Web API:n päätepisteelle. Search-päätepisteeltä haetaan käyttäjän syöttämän kappaleen tiedot. v1/artists/{id}/related-artists-päätepisteeltä haetaan syötetyn kappaleen esittäjän kanssa samanlaiset artistit. Näiden artistien suosituimmat kappaleet haetaan API-pyynnöillä v1/artists/{id}/top-tracks-päätepisteelle.

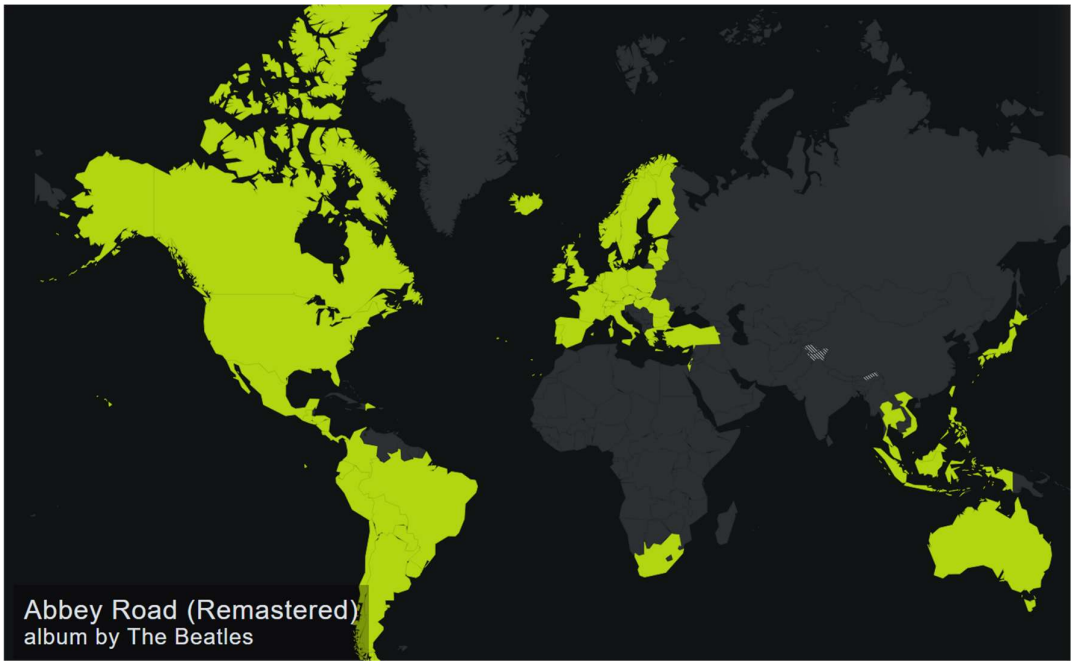
Kirjautumisessa sovellus hyödyntää Spotify Web API:n valtuutuskoodi-valtuutusvirtaa. Sovellus pyytää lupaa yhteensä seitsemään käyttöoikeuslohkoon. Nämä yhdessä antavat sovellukselle seuraavat oikeudet:

- lukuoikeus käyttäjän sähköpostiosoitteeseen
- muokkausoikeus julkisiin ja yksityisiin soittolistoihin
- lukuoikeus käyttäjän musiikkikirjaston sisältöön
- lukuoikeus käyttäjän kuunnelluimpiin artisteihin ja kappaleisiin

- luku- ja muokkausoikeus käyttäjän seuraamiin artisteihin ja käyttäjiin. [15.]

4.2 Album Availability

Album Availability on yhden sivun web-sovellus, joka esittää nimensä mukaisesti Spotify-julkaisujen saatavuuden graafisesti. Jokaiselle Spotify-julkaisulle on määritelty maat, joissa se on kuunneltavissa. Album Availabilityn avulla käyttäjä voi tarkistaa, missä maissa mikäkin julkaisu on kuunneltavissa Spotifyssa. Sovellus esittää tuloksen graafisesti maailmankartan avulla. Maailmankartassa ovat korostettuna ne maat, joissa julkaisu on kuunneltavissa. Kuvassa 9 on nähtävissä sovelluksen luoma maailmankartta maista, joissa The Beatlesin Abbey Road -albumi on kuunneltavissa.



Kuva 9. The Beatles -yhtyeen albumi Abbey Roadin saatavuus esitettynä kartan avulla Album Availability -sovelluksessa. Albumi on kuunneltavissa Spotifyssa korostetuissa maissa.

Sovellus hyödyntää Spotify Web API:n albums-päätepistettä metatietojen hakemisessa. Albums-päätepisteen palauttama JSON-olio sisältää avaimen ”available_markets”, jonka arvo on maakoodilista. Nämä maakoodit vastaavat maita, joissa julkaisu on kuunneltavissa.

Album Availability vaatii käyttäjältä valtuuttamista ennen API-pyynnön tekemistä. Sovellus hyödyntää epäsuora myöntäminen-valtuutusvirtaa. Sovellus ei hyödynnä käyttäjätilin

tietoja toiminnassaan, joten kirjautumisen ainoa tarkoitus on hakea käyttöoikeustietue API-pyyntöä varten. [16.]

4.3 Spotlistr

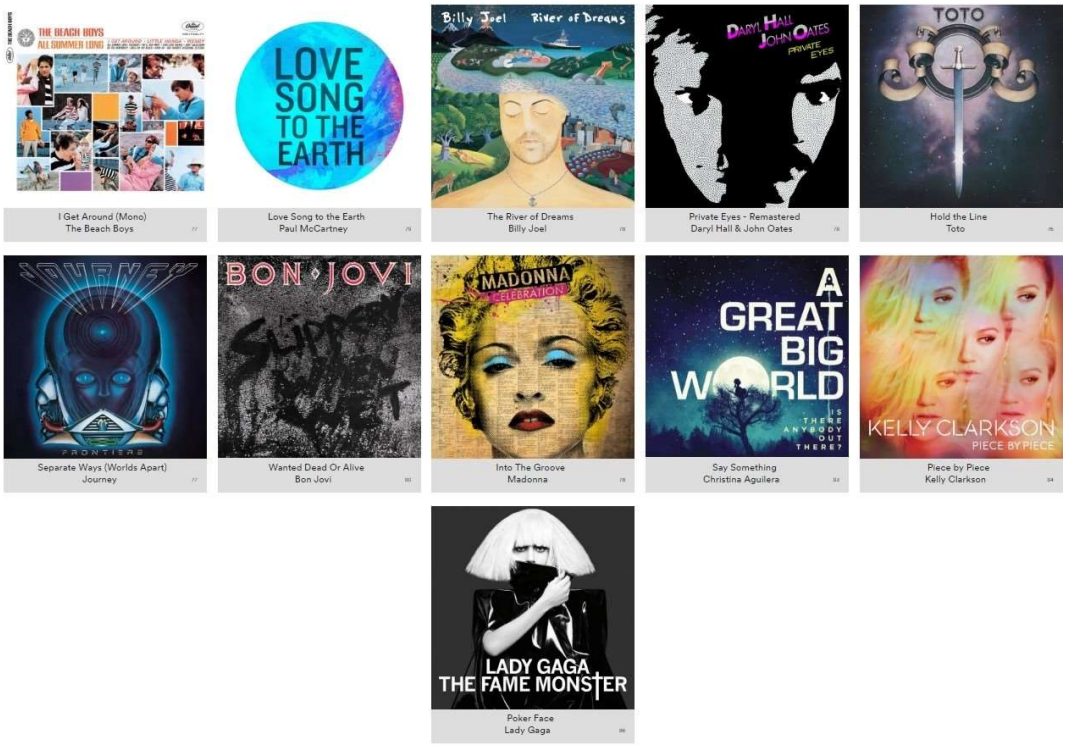
Spotlistr on Bob Niscon kehittämä web-pohjainen soittolistageneraattori. Sovellus kykenee muuntamaan Youtube- ja Soundcloud-soittolistat Spotify-soittolistoiksi sekä generoimaan kokonaan uusia soittolistoja muiden palveluiden rajapintoja hyödyntäen.

Spotlistr vaatii käyttäjää kirjautumaan sisään Spotify-tunnuksellaan ja luovuttamaan sovellukselle käyttöoikeuden tunnuksen tietoihin ennen kuin sen toiminnallisuudet ovat käytettävissä. Sovellus hyödyntää valtuutuskoodi-valtuutusvirtaa. Valtuutuksen yhteydessä Spotify Web API luo käyttöoikeustietueen, jota sovellus käyttää myöhemmin API-pyynnöissä. Yhteensä neljä Spotify Web API:n käyttöoikeuslohkoa hyväksytetään käyttäjällä kirjautumisen yhteydessä. Nämä antavat sovellukselle oikeuden lukea ja muokata käyttäjän julkisia sekä yksityisiä soittolistoja.

Sovellus käyttää Spotify Web API:a kappaleiden etsimiseen (search-päätepiste), ja mikäli käyttäjä niin haluaa, hänen soittolistojensa muokkaamiseen (v1/users/{user_id}/playlists/{playlist_id}/tracks-päätepiste). Sen lisäksi sovellus hyödyntää myös Last.fm:n, Soundcloudin, Youtuben ja Redditin API:a datan hakemisessa ja soittolistojen luomisessa. [17.]

4.4 Boil the Frog

Boil the Frog on Paul Lameren luoma web-sovellus, joka luo Spotify-soittolistan käyttäjän syöttämien kahden artistin perusteella. Sovelluksen tarkoituksena on luoda yhtenäinen soittolista minkä tahansa kahden artistin välille niin, ettei muutos seuraavan ja edellisen kappaleen välillä missään vaiheessa soittolistaa ole liian äärimmäinen, vaan kappale kappaleelta siirrytään lähemmäs kohdeartistia esimerkiksi genren ja energisyyden puolesta. Kuvassa 10 on nähtävissä Boil The Frogin luoma soittolista The Beach Boysin ja Lady Gagan pohjalta. Soittolistan ensimmäinen kappale on The Beach Boysin ”I Get Around”, josta siirrytään kappale kappaleelta musiikillisesti lähemmäs Lady Gagaa, jonka kappale ”Poker Face” on soittolistan viimeinen kappale.



Kuva 10. Boil The Frogin luoma soittolista The Beach Boysista Lady Gagaan.

Sovellus on täysin selainpohjainen eikä vaadi palvelinta toimiakseen. Kaikki sovelluksen toiminnot suoritetaan käyttäjän selaimessa. Käyttäjän syöttäessä haluamansa kaksi artistia sovellukseen, tämän selain tekee API-pyynnön Smarter Playlists-web-sovellukselle. Myös Smarter Playlists on Paul Lameren luoma web-sovellus. Vastauksena pyyntöön Smarter Playlists palauttaa JSON-olion, joka sisältää lähimmän ”reitin” lähtöartistista kohdeartistiin. Käytännössä tämä tarkoittaa artistilistaa, jonka joka kohdassa ero seuraavan ja edellisen artistin välillä on musiikillisesti mahdollisimman pieni. JSON-olio sisältää myös jokaisen listassa olevan artistin kymmenen suosituinta kappaletta Spotifyssa. Boil the Frog arpoo jokaisen artistin kohdalla näistä kymmenestä kappaleesta yhden, joka valitaan generoitavaan soittolistaan.

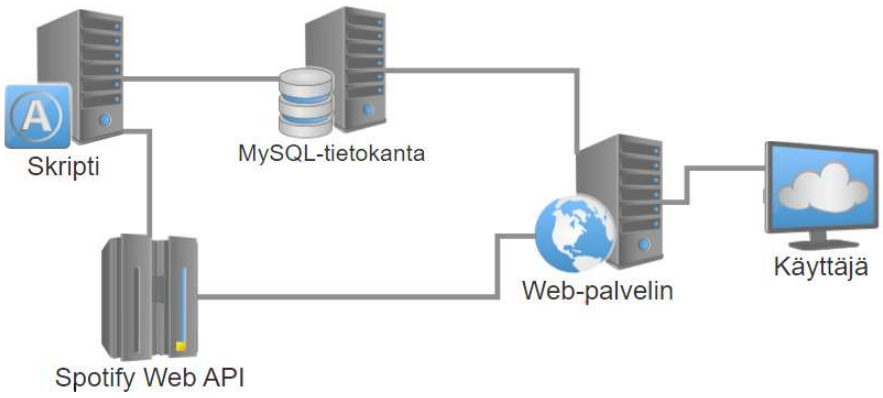
Kun soittolista on luotu, tarjoutuu käyttäjälle mahdollisuus tallentaa se Spotifyssa. Tämä edellyttää, että käyttäjä kirjautuu Spotify-tunnuksellaan sisään sovellukseen. Koska sovellus on täysin selainpohjainen, käytetään epäsuora myöntäminen-valtuutusvirtaa, joka ei vaadi lainkaan palvelinpuolen koodia. Sovellus pyytää lupaa yhteensä kolmeen käyttöoikeuslohkoon: playlist-readiin, playlist-modifyyn ja playlist-modify-privateen. Näistä muodostuvan näkyvyysalueen avulla sovelluksella on lukuoikeus käyttäjän julkisiin soittolistoihin ja muokkausoikeus käyttäjän julkisiin sekä yksityisiin soittolistoihin. Sovellus

tarvitsee muokkausoikeuden tallentaakseen luodun soittolistan käyttäjän tunnukselle. [18.]

5 Sube-sovellus

Työtä varten toteutettiin myös Spotify Web API:a hyödyntävä web-sovellus. Sovellusta kutsutaan tässä työssä nimellä Sube. Suben tarkoituksena on tarjota käyttäjälle selkeä ja automaattisesti päivittyvä lista uusista Spotify-julkaisuista määrätyiltä artisteilta. Sovellus seuraa tällä hetkellä 537 artistia. Siinä missä aiemmassa luvussa esitellyt sovellukset räätälöivät soittolistoja, Suben pääasiallinen tehtävänä on tuoda käyttäjän tietoon uusi musiikki seuratuilta artisteilta.

Sube rakentuu Node.js-web-palvelimesta, Python-skriptistä ja SQL-tietokannasta. Kuvassa 11 on havainnollistettu sovelluksen arkkitehtuuri. Web-palvelin tarjoaa sovelluksen käyttöliittymän. Python-skripti hakee uudet julkaisut Spotifyn API:lta ja lisää ne SQL-tietokantaan. Web-palvelin hakee uusimmat julkaisut SQL-tietokannasta, kun käyttäjä saapuu sivustolle.



Kuva 11. Verkkokaavio Suben arkkitehtuurista. Python-skripti hakee dataa Spotify Web API:lta ja lisää sen MySQL-tietokantaan. Käyttäjän saapuessa web-sivustolle web-palvelin hakee datan MySQL-tietokannasta sekä Spotify Web API:lta.

Python-skripti hakee päivittäin Spotify Web API:lta uusimmat julkaisut artisteilta, joita se on määrätty seuraamaan. Mikäli API palauttaa uudemman julkaisun kuin edellisellä skriptin ajolla, lisätään julkaisun metadata SQL-tietokantaan.

5.1 Suunnittelu ja idea

Idea sovellukselle syntyi henkilökohtaisesta tarpeesta Spotifyn käyttäjänä. Spotify tarjoaa uuden musiikin löytämiseen kaksi viikottain generoituvaa soittolistaa sekä uudet julkaisut-listan, joka sisältää vain muutamia hiljattain lisättyjä julkaisuja. Soittolistat kulkevat nimillä ”Discover Weekly” ja ”Release Radar”. Ensimmäiseen näistä Spotifyn algoritmit valitsevat kappaleita, joiden arvioidaan olevan käyttäjän mieleen. Toiseen soittolistaan valikoidaan uusia kappaleita artisteilta, joita käyttäjä on kuunnellut tunnuksellaan aiemmin. Molemmat soittolistat sisältävät 30 kappaletta. Nämä tavat löytää uutta musiikkia riittävät todennäköisesti valtaosalle Spotifyn käyttäjäkunnasta, mutta aktiivikäyttäjällä uusi kuunneltava loppuu pian kesken.

Itse Spotifyn aktiivikäyttäjänä lähdin tutkimaan, olisiko Spotify Web API:n avulla mahdollista toteuttaa sovellus, jolla uudet julkaisut kaikilta haluamiltani artisteilta löytäisi tyhjentävästi yhdestä paikasta. Spotify Web API:a tutkimalla kävi selväksi, että kattavan albumi-metadatan ansiosta seurantasovellus olisi API:n avulla mahdollista toteuttaa.

Ensimmäisenä täytyi selvittää, miltä päätepisteiltä tarvittavan datan saa haettua. Tavoitteena oli mahdollisimman vähäisellä API-kutsujen määrällä saada selville jokaisen ennalta määritetyn artistin viimeisin julkaisu Spotifyssa. v1/search-päätepiste osoittautui erittäin päteväksi: sen avulla pystyi yhdellä kutsulla pyytämään metadataa jopa 50 julkaisusta liittämällä artistin nimen API-pyynnön query-kenttään. Valitettavasti tämän päätepisteen palauttama metadata julkaisuista oli varsin suppeaa eikä sisältänyt dataa julkaisupäivästä, joten lisäpyyntöjä täytyi tehdä.

Kun julkaisujen ID:t oli poimittu ensimmäisen pyynnön vastauksesta, piti selvittää näiden julkaisupäivät. Tämä onnistui v1/albums-päätepisteen avulla. Päätepiste ottaa vastaan jopa 20 albumin ID:t yhdessä API-pyynnössä ja palauttaa jokaisesta julkaisusta enemmän metadataa kuin v1/search-päätepiste. Metadatan joukossa oli myös julkaisupäivä. Kahden API-pyynnön jälkeen pystyttiin järjestämään minkä tahansa artistin Spotify-julkaisut julkaisupäivän perusteella. Alkuperäiseen tavoitteeseen päästiin siis vähimmillään kahdella API-kutsulla per seurattu artisti.

Seuraavaksi oli päätettävä ohjelmointikieli, jolla API-pyynnöt ja vastausten tallentaminen toteutettaisiin. Vaihtoehtoina olivat muun muassa Java, Python ja Javascript. Valitsin kieleksi lopulta Pythonin sen keveyden sekä sille löytyvien kirjastojen takia.

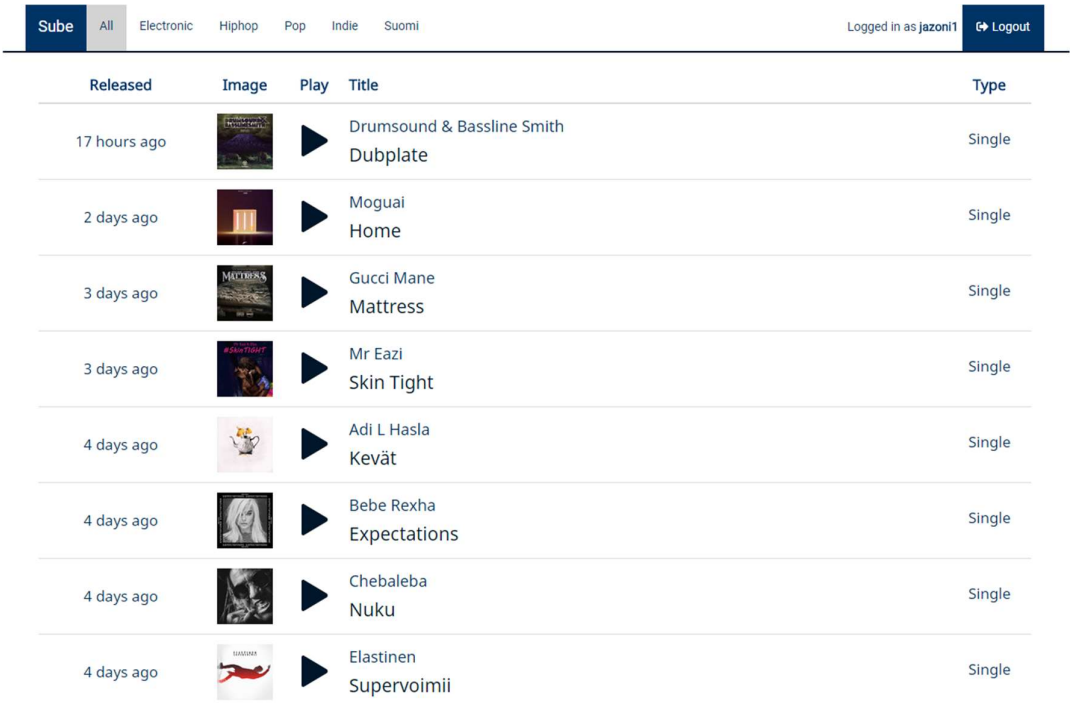
Tietokannaksi valitsin MySQL-relaatiotietokannan ja ohjelmistoksi HeidiSQL:n aiemman kokemuksen ja tuntemuksen perusteella.

5.2 Komponentit

Tässä luvussa perehdytään komponentteihin, joista Sube rakentuu. Näitä ovat web-sivusto ja web-palvelin, skripti sekä tietokanta. Web-palvelin tarjoaa web-sivuston, joka toimii suben käyttöliittymänä. Skriptin tehtävänä on hakea metadata uusista julkaisuista Spotify Web API:lta. Tietokantaan tallennetaan skriptin hakema metadata.

5.2.1 Web-sivusto ja -palvelin

Suben käyttöliittymänä toimii web-sivusto. Kaikkien sivujen yläreunassa on navigointipalkki, jonka avulla käyttäjä voi navigoida sivustolla. Navigointipalkki sisältää Koti-painikkeen lisäksi linkit useisiin listoihin. Listat sisältävät eri genrejen julkaisuja. Vaihtoehtoina ovat tällä hetkellä All-, Electronic-, Hiphop-, Pop-, Indie- sekä Suomi-listat. All-lista sisältää kaikki uudet julkaisut kaikista genreistä. Electronic-lista sisältää uudet julkaisut elektronisen musiikin saralta. Hiphop-listasta löytyy uusimmat hiphop-julkaisut ja niin edelleen. Suomi-lista sisältää uusimmat suomalaiset Spotify-julkaisut. Kuva 12 on ruutukaappaus all-listasivulta.



Kuva 12. Kuvakaappaus All-listasivun julkaisulistasta. All-lista sisältää nimensä mukaisesti uudet julkaisut kaikista genreistä.

Navigointipalkissa sijaitsee myös Login-painike, jota painamalla käyttäjä ohjataan Spotifyn kirjautumissivulle. Mikäli käyttäjä kirjautuu onnistuneesti sisään, ohjataan hänet takaisin Suben sivulle, jossa Login-painiketta alun perin painettiin. Navigointipalkin Login-painike muuttuu Logout-painikkeeksi ja sen vieressä näytetään Spotify-käyttäjätunnus, jolla kirjauduttiin sisään (kuva 6). Samalla URL:n perään on lisätty Spotifyn generoimat käyttöoikeustietue sekä päivitystietue. Kirjautumisen yhteydessä tietueille on myönnetty käyttöoikeuslohkot user-modify-playback-state ja user-read-playback-state, jotka antavat sovellukselle oikeuden lukea ja muokata käyttäjän toiston tilaa Spotifyn sovelluksissa. Tietueet poimitaan URL:sta ja lisätään käyttäjän selaimen paikalliseen varastoon (engl. local storage), josta ne poimitaan myöhemmin tarvittaessa. Paikallinen varasto on käyttäjän selaimessa sijaitseva tietovarasto, johon voidaan tallentaa sivustokohtaista dataa. Tietueet tuhotaan paikallisesta varastosta käyttäjän painaessa Logout-painiketta.

Etusivulla on lyhyt esittely web-sivustosta ja sen eri listoista. Sivuston muut sivut ovat listasivuja, joilla ovat eri genrejen listat. Listasivulla on taulukko, joka sisältää sivun nimeä vastaavan genren uusia julkaisuja. Julkaisut ovat taulukossa järjestetty niin, että uusin julkaisu on ylimpänä. Samana päivä julkaistut julkaisut ovat aakkosjärjestyksessä.

Taulukon rivit on jaettu viiteen sarakkeeseen: ensimmäisessä sarakkeessa on julkaisuaika, toisessa julkaisun kansikuva, kolmannessa soittopainike, neljännessä artistin sekä julkaisun nimi ja viimeisessä sarakkeessa julkaisun tyyppi. Soittopainikkeen toiminnallisuus riippuu käyttäjän kirjautumisen tilasta.

Jos käyttäjän selaimen paikallisesta varastosta löytyy käyttöoikeustietue, tehdään käyttäjän painaessa soittopainiketta API-pyyntö AJAX:lla Spotify Web API:n v1/me/player/play-päätepistelle. Pyynnön query-kentässä lähetetään sen julkaisun id, jonka riviltä soittopainiketta painettiin. Julkaisu lähtee soimaan käyttäjän laitteessa, mikäli Spotify-sovellus on jollain laitteella auki.

Mikäli käyttäjä ei ole kirjautunut sisään eikä käyttöoikeustietuetta löydy selaimen paikallisesta varastosta, avautuu soittopainiketta painattaessa sivun oikeaan alakulmaan Spotifyn ”Play Button” -soitin. Play Button on Spotifyn tarjoama upotettu soitin, joka sijoitetaan iframe-elementin sisään web-sivustolle tai esimerkiksi blogiin. Soittimen avulla voi kuunnella Spotify-kappaleita web-selaimessa. Kappaleesta soitetaan 30 sekunnin esittelypätkä, mikäli käyttäjällä ei ole Spotify-sovellusta auki laitteessaan. Soittimeen ladataan se julkaisu, jonka riviltä käyttäjä painoi soittopainiketta.

Sivuston koodissa hyödynnetään useita kirjastoja ja kehyksiä. Käytettyjä kirjastoja ovat muun muassa jQuery, livestamp.js ja Font Awesome. jQueryä käytetään AJAX-pyyntöjen tekoon, livestamp.js muuntaa taulukon julkaisupäivän muotoon ”x päivää sitten” ja Font Awesome tarjoaa sivustolla käytettävät ikonit. Näiden JavaScript-kirjastojen lisäksi sivuston ulkonäössä hyödynnetään myös Bootstrap-kehystä, jonka ansiosta sivusto on täysin responsiivinen. Sivuston mallimoottorina (engl. template engine) toimii Pug, jonka avulla tietokannan sisältö saadaan haettua julkaisutaulukoihin.

Web-sivusto tarvitsee toimiakseen web-palvelimen. Palvelimen tehtävänä on vastata käyttäjien tekemiin HTTP-pyyntöihin ja tarjota web-sivusto, jolla Subein käyttöliittymä sijaitsee. Suben web-palvelimena toimii Node.js-palvelin.

Node.js on Chrome V8:n JavaScript-moottorin pohjalle rakennettu sovellusalusta. Se mahdollistaa JavaScriptin käyttämisen selainpuolen lisäksi myös palvelinpuolen ohjelmointikielenä. Node.js sopi Suben palvelimeksi loistavasti muun muassa sen keveyden ja skaalautuvuuden puolesta.

5.2.2 Skripti

Python-skripti on tämän sovelluksen sydän. Sen tehtävänä on hakea Spotify Web API:lta uudet julkaisut päivittäin ja tallentaa ne tietokantaan. Skripti sisältää myös listan seuratuista artisteista.

Skriptissä hyödynnetään kolmea Python-moduulia: spotipyä, json:ia, ja MySQLdb:tä. Spotipy on kevyt Python-kirjasto, jonka avulla API-kutsut Spotify Web API:lle yksinkertaistuu huomattavasti. Pythonin sisäisen json-moduulin avulla API:n palauttamia JSON-vastauksia pystytään jäsentelemään sujuvasti. MySQLdb-moduulin avulla Python-skripti pystyy kommunikoimaan MySQL-tietokannan kanssa.

Suorituksen alussa skripti valtuuttaa itsensä spotipy-moduulin avulla ja hankkii käyttöoikeustietueen Spotify Web API:lta. Saatu käyttöoikeustietue liitetään jokaiseen skriptin tekemään API-pyyntöön. Valtuutusvirtana käytetään asiakkaan valtuustiedot-virtaa, koska skriptin tekemät API-pyynnöt ovat aina vakioita eikä niissä hyödynnetä käyttäjäkohtaista dataa.

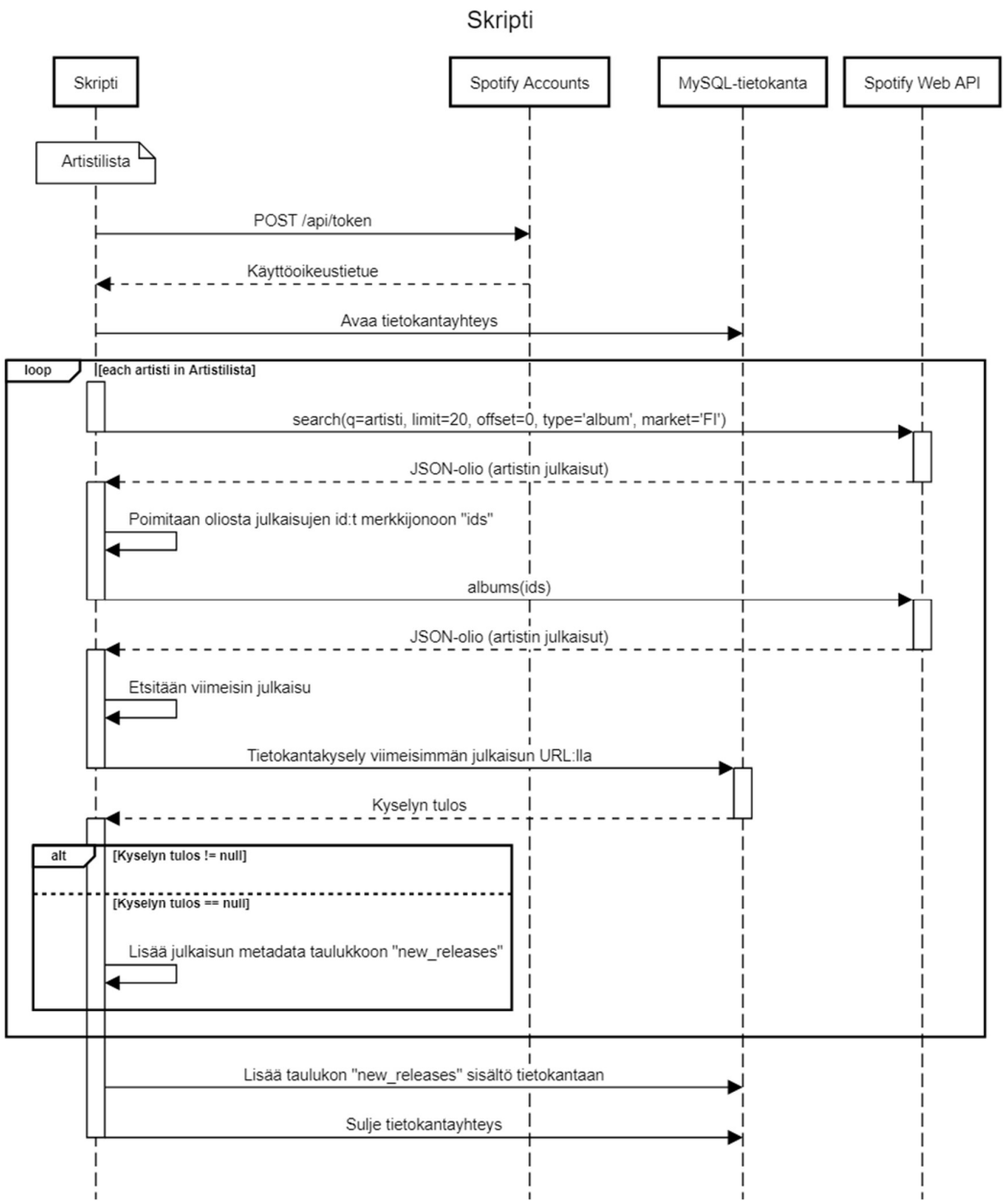
Valtuutuksen jälkeen skripti avaa yhteyden tietokantaan MySQLdb-moduulia käyttäen. Samalla luodaan kursori-olio, jonka avulla tietokantaan voidaan myöhemmin tehdä kyselyjä ja lisätä dataa.

Seuraavaksi skripti käy läpi artistilistan ja tekee jokaisen artistin kohdalla tietyt toimenpiteet. Ensimmäisenä tehdään Spotify Web API:n search-päätepisteelle pyyntö. Pyynnön ”type”-parametrin arvo on ”album” ja ”query”-parametrin arvo on artistin nimi. Näillä parametreillä API palauttaa query-kentässä olevan artistin julkaisujen metatiedot JSON-muodossa. Tämän jälkeen vastaus muutetaan JSON-olioksi, jotta sen eri osiin pystytään viittaamaan helposti.

JSON-oliosta poimitaan kaikkien sen sisältämien julkaisujen id:t ja ne lisätään Python-listaan. Listan alkioista, eli yksittäisistä id:istä kootaan merkkijono. Seuraavaksi skripti tekee toisen API-pyynnön, tällä kertaa albums-päätepisteelle. Albums-päätepiste ottaa vastaan listan id:itä, jotka välitetään sille merkkijonona URL:n perässä. Vastauksena skripti saa taas metatiedot näistä julkaisuista JSON-muodossa. Albums-päätepisteen palauttama metadata julkaisuista on kattavampaa kuin search-päätepisteen palauttama metadata. Kattavamman metadatan joukossa on myös julkaisupäivämäärä. Seuraavaksi täytetään uusi Python-hakemisto julkaisuilla niin, että kukin hakemiston alkio vastaa yhtä

julkaisua. Alkiot järjestetään avaimen ”release_date” mukaan niin, että hakemiston ensimmäinen alkio on artistin uusin julkaisu.

Kun artistin uusin julkaisu on selvillä, tarkistetaan, löytyykö se jo ennestään tietokannasta. Tämä on toteutettu poimimalla hakemiston ensimmäisestä alkiosta julkaisun uniikki URL ja vertaamalla sitä tietokannassa jo ennestään oleviin URL:hin. Skripti etsii tietokannasta riviä, jonka ”Linkki”-sarakkeessa oleva URL on sama kuin julkaisun URL. Jos yhtään riviä ei löydy, lisätään julkaisun tiedot uuteen tilapäiseen hakemistoon, jonka sisältö lisätään myöhemmin tietokantaan. Skriptin toiminta on havainnollistettu kuvassa 13 sekvenssikaavion avulla.



Kuva 13. Sekvenssikaavio Python-skriptin toiminnasta.

Skripti käy läpi jokaisen artistin listassa ja tekee edellä mainitut toimenpiteet. Kun lista on käyty läpi, lisätään tilapäisen hakemiston sisältö tietokantaan. Hakemiston sisältö koostuu ajon aikana löydetyistä uusista julkaisuista. Tietokantaan lisätään seuraavat tiedot julkaisusta: artistin nimi, julkaisun nimi, julkaisupäivämäärä, julkaisun URL, kansikuvan URL sekä albumityyppi.

Jotta käyttäjälle voidaan tarjota aina ajantasainen lista uusista julkaisuista, täytyy skriptiä ajaa tietyin väliajoin. Spotify julkaisee uudet julkaisut aina paikallista aikaa keskiyöllä.

Tästä johtuen skriptiä tarvitsee ajaa vain kerran päivässä. Skripti on ajastettu ajettavaksi joka yö kello 0:05. Ajon hoitaa palvelimella Windows Task Scheduler.

5.2.3 Tietokanta

Skriptin hakema metadata varastoidaan MySQL-relaatiotietokantaan. Sitä ylläpidetään samalla palvelimella kuin web-palvelinta. Tietokanta sisältää ainoastaan yhden taulun nimeltään biisit.

Taulussa on kuusi saraketta: Artisti, Kappale, Julkaistu, Linkki, Kuva sekä Tyyppi. Tietokantataulun rakenne on nähtävissä kuvassa 13. Artisti-sarakkeessa on artistin nimi (esim. Abba), Kappale-sarakkeessa julkaisun nimi (esim. Golden Hits), Julkaistu-sarakkeessa on julkaisupäivämäärä ja Linkki-sarakkeessa julkaisun URL. Kuva-sarake sisältää julkaisun kansikuvan URL:n ja Tyyppi-sarake julkaisun tyypin. Sarakkeista kaikki paitsi Julkaistu ovat tyyppiä VARCHAR. Julkaistu-sarake on tyyppiä DATE.

Artisti	Kappale	Julkaistu	Linkki	Kuva	Tyyppi
From Ashes to New	The Future	2018-04-20	**https://open.spotify.com/alb...**	https://i.scdn.co/image/c63...	album
drake	Way Over	2018-03-24	**https://open.spotify.com/alb...**	https://i.scdn.co/image/d6...	single
1017 Eskimo	Yeah Yeah (feat. Gucci Mane)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/c53...	single
Acoustic Heartstrings	AH Performs Mumford & Sons	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/2a...	album
Alle Farben	H.O.L.Y.	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/f7e...	single
Anitta	Vai malandra (feat. Tropkillaz & DJ Yuri...	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/03...	single
Anton Powers	Heart For Sale (Radio Edit)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/19...	single
Armin van Buuren	Sex, Love & Water (Sunnery James & ...	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/4d...	single
Autograf	Dead Soon	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/37...	single
BlocBoy JB	Rover 2.0 (feat. 21 Savage)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/8b...	single
BlocBoy JB	Rover 2.0 (feat. 21 Savage)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/ef2...	single
Blond	Psycho - Post Malone (Spanish Version)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/09f...	single
Breathe Carolina	Long Live House Music	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/a0f...	single
Bro Safari	Proper (Remixes)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/81c...	single
Bruno Mars	Finesse (Remixes) [feat. Cardi B]	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/e9...	single
BUNT.	For You My Love (feat. BEGINNERS)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/d0...	single
Burak Yeter	Crash	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/36...	single
Camo & Krooked	Mosaik (Remixes)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/35...	album
Chief Keef	The Leek (Vol. 5)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/be...	album
Chrizay	Solo	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/27...	single
CHVRCHES	Get Out (The Remixes)	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/6b...	single
Diplo	California	2018-03-23	**https://open.spotify.com/alb...**	https://i.scdn.co/image/3fb...	single

Kuva 14. Biisit-taulun rakenne. Taulu koostuu kuudesta sarakkeesta ja sisältää tietoa uusista Spotify-julkaisuista.

5.3 Jatkokehitys

Sovellus tarjoaa jatkokehityksen kannalta monenlaisia vaihtoehtoja. Tässä luvussa käydään läpi muutamia mahdollisia kehityskohtia.

Tällä hetkellä skripti hakee artistin julkaisut API-pyynnöllä search-päätepisteelle. Pyynnön query-kentässä on artistin nimi ja type-kentässä ’album’. Näiden parametrien avulla Spotifyn hakualgoritmi osaa palauttaa albumit, joiden artistin tai julkaisun nimessä on query-kentässä olleen artistin nimi. Aika ajoin tulee eteen tilanne, jossa täysin tuntematon artisti on julkaissut albumin, jonka nimessä on etsityn artistin nimi ja hakualgoritmi palauttaa tämän julkaisun. Tästä esimerkkinä artistilistassa on artisti nimeltään ”The Weeknd”, jonka uusinta julkaisua Sube yrittää selvittää search-päätepisteen avulla. Artisti nimeltään ”AXPMP” on julkaissut hiljattain singlen nimeltään ”The Weeknd Type Beat”. Suben skripti lisää näistä viimeisen tietokantaan, koska hakualgoritmi palauttaa sen uusimpana julkaisuna, joka sisältää merkkijonon ”The Weeknd”. Tämä johtaa tilanteeseen, jossa tietokantaan tallentuu ei-toivottuja julkaisuja tuntemattomilta artisteilta. Ongelma voidaan korjata joko muuttamalla käytettyjä päätepisteitä tai poimimalla julkaisun metadatasta kaikkien julkaisussa mukana olevien artistien nimet ja vertaamalla niitä haluttuun artistiin. Tämä on mielestäni tärkein kehityskohta tällä hetkellä.

Toinen kehityskohta Suben toiminnassa olisi lisäominaisuuksien lisääminen kirjautuneelle käyttäjälle. Tällä hetkellä käyttäjä ei hyödy kirjautumisesta muuta kuin mahdollisuuden aloittaa julkaisun toisto laitteessaan suoraan selaimesta. Laajemmalla näkyvyysalueella käyttäjälle voisi antaa mahdollisuuden esimerkiksi lisätä julkaisu haluamaansa soittolistaan tai kappalejonoon, mikäli Spotify julkaisee tulevaisuudessa päätepisteen, joka tämän mahdollistaa.

Artistilistan tuominen näkyviin web-sivustolle on myös yksi tulevaisuuden kehityskohdista. Käyttäjälle voisi myös antaa mahdollisuuden toivoa artistilistan laajentamista haluamillaan artisteilla, mikäli niitä ei löydy vielä seuratuista. Tämä onnistuisi esimerkiksi HTML-lomakkeella. Uusien genrejen sekä artistien lisääminen olisi myös yksi mahdollinen kehityskohta tulevaisuudessa.

6 Yhteenveto

Tämän työn tavoitteena oli perehtyä Spotify Web API:n käyttöön ja sen tarjoamiin mahdollisuuksiin web-sovelluskehityksessä. Tarkoituksena oli tutustua API:n arkkitehtuuriin ja käyttöön yleisellä tasolla sekä toteuttaa hankitun tiedon pohjalta API:a hyödyntävä web-sovellus.

Työn lähtökohtana tunsin Spotifyn palveluna hyvin, koska olin käyttänyt sitä vuosia. Spotifyn API:sta, tai API:en toiminnasta yleisesti sen sijaan en tiennyt juuri mitään ennen työtä. Lähdinkin liikkeelle hakemalla tietoa Spotify API:sta sekä tutustumalla REST:n periaatteisiin, joiden pohjalta Spotify on Web API:nsa rakentanut. API:iin perehdyttiin ensin sen tekniikan ja arkkitehtuurin kautta ja myöhemmin esiteltiin sen erilaisia käyttötapoja web-sovellusten muodossa.

Työn käytännön osuutena toteutettiin web-sovellus, joka Spotifyn API:a hyödyntäen hakee uudet julkaisut määrätyiltä artisteilta ja kokoaa näistä listan web-sivulle. Sivustolle toteutettiin myös Spotify-tunnuksella kirjautuminen, jonka avulla käyttäjä saa julkaisun soimaan laitteessaan suoraan sovelluksen web-sivustolta. Sovellusta suunnitellessa ja toteuttaessa opin paljon uutta paitsi Spotify Web API:sta, myös web-ohjelmoinnista ja palvelinohjelmoinnista. API:n dokumentaatio osoittautui työn aikana erittäin päteväksi, vaikka pieniä puutteita löytyikin.

Spotify Web API:n päätepisteitä tarkastelemalla havaittiin, että päätepisteet voidaan jakaa karkeasti kolmeen ryhmään: Spotifyn julkista katalogidataa (esim. musiikin metadata) palauttavat päätepisteet, käyttäjätilikohtaista dataa palauttavat ja muokkaavat päätepisteet sekä käyttäjän laitteen toiston tilaa muokkaavat ja palauttavat päätepisteet.

Spotify Web API:a hyödyntäviä web-sovelluksia tutkimalla havaittiin, että niitä löytyi internetistä kohtalaisen vähän verrattuna esimerkiksi sosiaalisen median palvelujen rajapintoja hyödyntäviin sovelluksiin. Suurin osa löytyneistä sovelluksista käytti Spotifyn API:a erilaisten soittolistojen luomiseen, eikä kovinkaan luovia käyttötapoja löytynyt. Yksi syy tähän voi olla se, ettei musiikin metadatalle lähtökohtaisesti ole laajaa kysyntää sovelluskehityksessä ja monet sitä hyödyntävät sovellukset ovat loppukäyttäjälle melko tarpeettomia. Spotify on myös itse tehnyt monet API:a hyödyntävät sovellukset tarpeettomiksi tuomalla palveluunsa sovellusta vastaavan toiminnallisuuden. Tällaisia ovat esimerkiksi käyttäjälle automaattisesti räätälöityvät soittolistat, joiden avulla uutta musiikkia on jatkuvasti tarjolla käyttäjälle.

Työssä saavutettiin sille asetetut tavoitteet, ja Spotify Web API tuli minulle henkilökohtaisesti erittäin tutuksi. Työn avulla saatiin selville, miten Spotifyn API:a voidaan hyödyntää web-sovelluksissa ja millaisia mahdollisuuksia se tarjoaa näille. Työn tuloksena syntyi myös Sube-sovellus, joka on oiva työkalu uuden musiikin löytämiseen Spotifyn omien

menetelmien ohelle. Aion jatkaa sovelluksen kehitystä myös jatkossa, ja seuraavia kehityskohteita ovat muun muassa kattavammat genrerajaukset sekä artistilistan tuonti näkyville sivustolle.

Lähteet

1 Tervonen, Kari. 2017. Musiikin kulutus ikäryhmittäin: mitä kuunnellaan, mistä tykätään – ja miksi?. Verkkojulkaisu. <https://www.teosto.fi/sites/default/files/files/Suomessa_ika_musiikin_kulutuksen_ylivoimainen_ykkosselittaja.pdf> Luettu 15.3.2018.

2 About, Spotify Press. 2017. Verkkoaineisto. <https://press.spotify.com/fi/about/> Luettu 15.3.2018.

3 Fielding, Roy Thomas. 2000. Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST). Verkkoaineisto. <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm> Luettu: 20.2.2018.

4 McIntery, Hugh. 2016. Spotify's Revenues Hit $2 Billion in 2015, But Losses Grow As Well. Verkkoaineisto. <https://www.forbes.com/sites/hughmcintyre/2016/05/24/spotifys-revenues-hit-2-billion-in-2015-but-losses-grow-as-well> Luettu: 15.3.2018.

5 United States Securities and Exchange Comission. 2018. Spotify Technology S.A. Form F-1. Verkkoaineisto. <https://www.sec.gov/Archives/edgar/data/1639920/000119312518063434/d494294df1.htm> Luettu: 1.3.2018.

6 Mulligan, Mark. 2017. Amazon Music: The Dark Horse Comes Out Of The Shadows. Verkkoaineisto. <https://www.midiaresearch.com/blog/amazon-music-the-dark-horse-comes-out-of-the-shadows> Luettu: 1.3.2018.

7 Steele, Anne. 2018. Apple Music on Track to Overtake Spotify in U.S. Subscribers. Verkkoaineisto. <https://www.wsj.com/articles/apple-music-on-track-to-overtake-spotify-in-u-s-subscribers-1517745720> Luettu: 1.3.2018.

8 Jussilainen, Mikko. 2015. Rest-pohjaisen web servicen kehittäminen. Verkkoaineisto. <https://www.theseus.fi/bitstream/handle/10024/102841/Jussilainen_Mikko.pdf> Luettu: 10.4.2018.

9 Web API User Guide, Spotify Developer. 2016. Verkkoaineisto. <https://developer.spotify.com/web-api/user-guide> Luettu: 10.4.2018.

10 Fielding, Roy Thomas. 2000. Fielding Dissertation: CHAPTER 3: Network-based Architectural Styles. Verkkoaineisto. <https://www.ics.uci.edu/~fielding/pubs/dissertation/net_arch_styles.htm#sec_3_4_1> Luettu: 10.4.2018.

11 Web API Endpoint Reference, Spotify Developer. 2016. Verkkoaineisto. <https://developer.spotify.com/web-api/endpoint-reference/> Luettu: 10.4.2018.

12 Get a Track, Spotify Developer. 2016. Verkkoaineisto. <https://developer.spotify.com/web-api/get-track/> Luettu: 10.4.2018.

13 Web API: Using Scopes, Spotify Developer. 2016. Verkkoaineisto. <https://developer.spotify.com/web-api/using-scopes/> Luettu: 10.4.2018.

14 Web API Authorization Guide, Spotify Developer. 2016. Verkkoaineisto. <https://developer.spotify.com/web-api/authorization-guide/> Luettu: 10.4.2018.

15 MagicPlaylist. 2015. Verkkoaineisto. <https://magicplaylist.co> Luettu: 16.4.2018.

16 Album Availability. 2015. Verkkoaineisto. <https://kaaes.github.io/albums-availability> Luettu: 16.4.2018.

17 Spotlistr. 2015. Verkkoaineisto. <http://spotlistr.herokuapp.com/> Luettu: 16.4.2018.

18 Boil The Frog. Verkkoaineisto. <http://static.echonest.com/BoilTheFrog/> Luettu: 16.4.2018